



knowledge_transformer

?

1. ?

1. ? YAML / ?
2. ? API ? appid/key ? CLI ?
3. ? HTTP ?
4. ?
5. ? Tracing ID ? Prometheus ? Flower ?
6. ? Celery ? Webhook ? Result Backend ? URL ? API ?

2. ?

? Office ?
REST API ? Celery ? FastAPI ? task_id ?
Celery ? Worker ? LibreOffice / Inkscape / FFmpeg ?
? Result Backend ? Webhook ?
? Celery Pipeline ? Celery ? Chain/Group/Chord ?

3. ?

- ?
sequenceDiagram participant API as API ? participant Q as Redis ?/
Result Backend participant W as Celery Worker participant P as ?
participant S as ? participant CB as ? API->>API: ?

```

task_id API->>Q: conversion.handle_batch Q-->>W: W-
>>S: URL W->>P: P-->>W: W-
>>S: converted/task_id/ W->>API: results[] alt
callback_url API->>CB: Webhook else API-->>API:
end

```

- i. task_id
- ii. conversion.handle_batch Redis
- iii. Worker / URL
- iv. converted/{task_id}/
- v. results[] callback_url

•

```

sequenceDiagram participant C as participant API as API
participant R as Redis /Result Backend participant W as Celery
Worker participant M as participant Mon as Flower/Prometheus
C->>API: POST /api/v1/convert API-->>C: 202 Accepted + task_id API-
>>R: conversion.handle_batch R->>W: W->>M: /
W-->>R: alt callback_url W->>C: Webhook
else / C->>R: C->>M: end Mon-->>API:
Mon-->>W:

```

- i. Pipeline Celery /api/v1/convert API Redis
- ii. API HTTP 202 task_id ID Result Backend
- iii. Celery Worker Webhook Result Backend Webhook Result Backend
- iv. Flower Prometheus API/Worker

4. ? ? ? ?

• Celery

- priority prefetch_multiplier worker task_time_limit soft_time_limit acks_late

- `API` `files[]` `SLA` `priority` `conversion.handle_batch` `Worker` `max_tasks_per_child` `batch_size`

❖❖❖❖

```
function submit_request(files, metadata):
    priority = calc_priority(files, metadata.sla)
    batch = split_files(files, metadata.batch_size)
    for chunk in batch:
        payload = build_payload(chunk, priority, metadata)
        push_to_queue(select_queue(chunk.plugin), payload)

worker_loop():
    settings = load_runtime_limits()
    while worker_alive():
        task = fetch_from_queue(settings.prefetch)
        if not task:
            continue
        with deadline(settings.task_time_limit):
            try:
                for file in task.files:
                    artifact = run_plugin(file)
                    upload_to_storage(task.task_id, artifact)
                ack(task)
            except TimeoutError:
                mark_failed(task, "timeout")
                requeue_remaining(task)
            except Exception as exc:
                mark_failed(task, str(exc))
                maybe_retry(task)
```

- ❖❖❖❖❖
 - `LibreOffice headless soffice` `Office→PDF/HTML/` `API` `URL` `source_format` `filter`

```
--convert-to pdf:writer_pdf_Export ①
tmpdir ② soffice
--headless --norestore --nodefault ③
stderr ④
tmpdir
```

- `/Inkscape CLI SVG/EPS/PDF PNG/JPEG/WebP ① --export-area-drawing ② dpi ③ Inkscape --actions`

■

- `PDF/EPS Inkscape SVG`
-
- `target_format PNG/JPEG/WebP DPI /tmp/rag_converter/<task>`
- `results[]`
- `converted/{task_id}/`

flowchart LR
 A[SVG/EPS/PDF] --> B[
/
]
 B --> C[PNG/JPEG/WebP]
 C --> D[
/
]
 D --> E[converted/
task_id/]
 E --> F[]

- `FFmpeg source_format / target_profile Preset CRF ABR`

■

- `webm→mp4 MP4`

x264 Baseline + CRF 23
AAC 128kbps

- b. wav→mp3
16kHz/
c. mov→mp4
H.265 H.264

d. gif→mp4/png
+ scale MP4
PNG

5. lib






































































































- rag_converter.plugins
 ConversionInput/ConversionResult
 source_format target_format input_path / input_url
 / object_key metadata
 output_path object_key metadata

ConversionPlugin.convert(payload)	payload: ConversionInput
ConversionPlugin.describe()	
PluginRegistry.register(plugin_cls)	plugin_cls: Type[ConversionPlugin]
PluginRegistry.get(source, target)	source:str , target:str
PluginRegistry.list()	
load_plugins(module_names=None)	module_names: Iterable[str]
read_plugin_module_file(path)	path: str

[illegible]

🔗/🔗	🔗
POST /api/v1/convert	Headers: X-Appid , X-Key 🔗 Body: task_name , priority , callback_url , storage{endpoint,access_key,secret_key,bucket} 🔗🔗🔗🔗🔗 files[] , target_format , input_url / object_key / base64_data 🔗🔗🔗 files[]
GET /api/v1/formats	Headers: X-Appid , X-Key
GET /api/v1/monitor/health	Headers: X-Appid , X-Key
GET /healthz	🔗
Celery 🔗 conversion.handle_batch	payload = {task_id, files[], priority, callback_url, storage}
Pipeline submit_conversion_chain	files[] , priority
Pipeline submit_conversion_group	file_groups[][] , priority
Pipeline submit_conversion_chord	file_batches[][] , priority

7.    

-     
 -         status=failed + reason  
 - task_time_limit=300s        
 -             
 -              
-     
 -      +      
 - Webhook       
 - Result Backend/         
 - Pipeline Celery                  

8.  

- Celery Chain
- appid/key
- Redis LibreOffice Inkscape FFmpeg
- config/settings.yaml

9.     

???	???	??/???	????
doc , docx , ppt , xls	docx , pdf , html	LibreOffice soffice ??	????????????????????
svg , eps , pdf	png , jpeg , webp	Inkscape CLI	??/?????????????
gif , webp	png , mp4	GIF/WebP ?? + FFmpeg	????????????????
wav , flac , ogg , aac	mp3	FFmpeg Audio ??	????????????????
avi , mov , mkv , webm , mpeg	mp4	FFmpeg Video ??	????????????????
?????	?????	??/?????	????? CAD→PDF? AI→SVG ??

10.    

- Worker /tmp/rag_converter/<task> converted/{task_id}/... bucket

endpoint=http://localhost:9000

access_key=minioadmin

secret_key=minioadmin bucket=qadata

Redis Result Backend

API payload

storage.endpoint/access_key/secret_key/bucket

- Docker/K8s/ CPU

11.

- config/settings.yaml Redis/

file_limits	default_max_size_mb , per_format_max_size_mb , max_files_per_task	
logging	level , log_dir , max_log_file_size_mb , backup_count	
monitoring	prometheus_port , metrics_interval_sec , health_api	
minio	endpoint , access_key , secret_key , bucket , timeout	http://localhost:9000 / minioadmin / minioadmin / qadata
convert_formats	source , target , plugin	
api_auth	required , app_secrets_path ,	API

???	????	??
	header_appid	
celery	broker_url , result_backend , task_time_limit_sec , prefetch_multiplier	??????????
rate_limit	enabled , interval_sec , max_requests	API ?????
??	service_name , plugin_modules_file ?	????????????????

- ???? RAG_ ???? YAML ??
?? RAG_REDIS_URL ? RAG_MINIO_ENDPOINT ? RAG_API_AUTH_REQUIRED ?

????	????	??
RAG_REDIS_URL	celery.broker_url/result_backend	?? Celery Bro Result Backend
RAG_MINIO_ENDPOINT	minio.endpoint	????? Docker ?????
RAG_API_AUTH_REQUIRED	api_auth.required	????? API ???
RAG_FILE_LIMIT_MAX_SIZE	file_limits.default_max_size_mb	?????????
RAG_PROM_PORT	monitoring.prometheus_port	??? Promethe ???
?? RAG_*	?????→?????	?? pydantic-setti ????? K Compose ???

- CLI/ `manage_plugins.sh` `make_key.sh`

<code>manage_plugins.sh</code>	<code>config/plugins.yaml</code>	<code>list/install/remove</code>
<code>make_key.sh</code>	<code>API</code> <code>appid/key</code> <code>secrets/appkeys.json</code>	<code>--appid</code> <code>ID</code>
<code>docker-start.sh / stop.sh</code>	<code>Docker</code> <code>Compose</code>	<code>.env</code> <code>docker compose up/down</code>
<code>show_server.sh</code>	<code>API/Worker/</code> <code>/Redis</code>	

12.

- `start_server.sh` `FastAPI` `Celery Worker` `Flower`

FastAPI	<code>uvicorn rag_converter.app:app --host 0.0.0.0 --port \${API_PORT}</code>
Celery Worker	<code>celery -A rag_converter.celery_app.celery_app worker -l \${CELERY_LOG}</code>
Flower	<code>celery -A rag_converter.celery_app.celery_app flower --port \${FLOWER}</code>

- Celery Worker `Pipeline Celery` `.env`
`CELERY_BROKER_URL` `CELERY_RESULT_BACKEND` `MINIO_*`

??	?????	??
Worker	CELERY_BROKER_URL , CELERY_RESULT_BACKEND , CELERY_DEFAULT_QUEUE , CELERYD_PREFETCH_MULTIPLIER , TASK_TIME_LIMIT , MINIO_*	?? Broker/Backend??/?? ??????????
Pipeline Celery	BROKER_URL , RESULT_BACKEND , MINIO_ENDPOINT , MINIO_ACCESS_KEY , MINIO_SECRET_KEY , PIPELINE_QUEUE	?? Redis????????????????

13. ?????

??	????/??	??
FastAPI	≥ 0.104	?? REST API ???
Celery	≥ 5.3	?????
Redis	≥ 7	?? Celery Broker/Result Backend?
????	S3 ? MinIO?	????/????
LibreOffice	?? LTS	?????
Inkscape	≥ 1.3	?????
FFmpeg	≥ 5.0	????/??
Flower	≥ 1.2	Celery ?? UI?
Prometheus	≥ 2.46	?????

- Python ??? pyproject.toml ? Pipeline ??? requirements.txt ??
 README ?

14.      

- `pytest + HTTPX` Celery test worker mock
- `/` Webhook Pipeline Chain/Group/Chord
-

🔍🔍🔍	🔍🔍🔍 → 🔍🔍🔍🔍	🔍🔍 (MB)	🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍
doc_sample_small.doc	doc → docx	0.05	🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍 Chain🔍🔍
doc_sample_pdf.doc	doc → pdf	0.05	🔍🔍🔍🔍🔍 PDF🔍🔍🔍🔍 soft
html_inline_base64.json	html(base64) → pdf	0.01	🔍🔍🔍🔍🔍🔍🔍base64_d 🔍🔍🔍🔍🔍🔍🔍
ppt_marketing.ppt	ppt → pdf	48	🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍
svg_logo.svg	svg → png	0.8	🔍🔍/🔍🔍🔍🔍🔍 + Pipelin
gif_banner.gif	gif → mp4	25	🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍
webp_large.webp	webp → png	15	🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍
wav_podcast.wav	wav → mp3	180	🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍
flac_archive.flac	flac → mp3	220	🔍🔍🔍🔍🔍🔍🔍🔍 FILE_TO
mov_trailer.mov	mov → mp4	480	🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍 500 Pipeline Chord🔍
mkv_fail.mkv	mkv → mp4	300	🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍🔍
invalid_format.bin	bin → docx	1	🔍🔍🔍🔍🔍/🔍🔍🔍🔍🔍🔍🔍🔍🔍 🔍🔍🔍🔍🔍
auth_test.docx	docx → pdf	2	API key 🔍🔍🔍🔍🔍🔍🔍🔍
webhook_payload.json	-	-	Webhook 🔍🔍🔍🔍🔍🔍 5

- **Mock**

???	HTTP ??	???	??	??????
ERR_AUTH_INVALID	401	4011	?????appid ? key ??	????? key ?? key?
ERR_FILE_TOO_LARGE	400	4201	??????	?? per_format_max_ flac_archive.fi
ERR_BATCH_LIMIT_EXCEEDED	400	4202	?????/ ?????	?? max_files_ max_total_uploa
ERR_FORMAT_UNSUPPORTED	400	4203	??????	?????? invalid_format
ERR_TASK_FAILED	500	5001	??????	?????/ ??????